METHOD FOR GENERATING INTERRUPT COMMANDS IN A MICROPROCESSOR SYSTEM AND RELATIVE PRIORITY INTERRUPT CONTROLLER

Field of the Invention

[0001] The present invention relates in general to microprocessor systems, and in particular, to a method for generating interrupts in a microprocessor system and a corresponding interrupt controller with automatically incremented priority values as a function of the latency time and of the time limit within which the relative interrupt is to be processed.

Background of the Invention

[0002] During the functioning of a processor, it may be necessary to interrupt the execution of a program for carrying out particular instructions. This is done by way of signals called interrupts. An interrupt controller receives these signals, and depending on the received interrupt, sends to the microprocessor an interrupt command and an interrupt vector that specifies the memory address where an interrupt service routine (ISR) to be run is stored.

[0003] The microprocessor stops the operation in progress, saves the state of the program that was executing so that it may be resumed later, and carries out the instructions of the respective ISR based upon the received interrupt. When the ISR finishes, the

microprocessor restores the state of the program, and if there is not any other pending interrupt, resumes its execution from the point at which it had been interrupted.

[0004] Interrupt controllers commonly have priority registers that allow them to establish which interrupt among the many received and pending interrupts is to be processed first. A basic architecture of a known priority interrupt controller is depicted in Figure 1. The interrupts INT0, ..., INTm coming from peripherals are loaded in a pending interrupt register INT PENDING REG.

[0005] The circuit block IRQ MASK AND PRIORITY LOGIC comprises both the interrupts mask and a priority logic circuitry that receives an interrupt together with its priority level provided by the dedicated registers PRIORITY REGISTERS. The priority logic generates an interrupt request signal IRQ REQ and stores the relative priority HIGHEST PRIORITY INT in the register CURR IRQ PRIORITY REG.

[0006] The dashed perimeter delimits the circuit that processes the interrupt request signal and its priority HIGHEST PRIORITY INT. A state machine IRQ SM forms the core of the controller that receives the interrupt request signal and sends an interrupt command nIRQ to the processor. The interrupt request signal IRQ REQ selects an interrupt vector IRQ VECTOR corresponding to the required interrupt read from an interrupt table IRQ VECTOR REG containing interrupt vectors identifying ISR routines.

[0007] Figure 1 shows registers CURR IRQ PRIORITY REG and PRIORITY STACK used for managing the nested interrupts. The register CURR IRQ PRIORITY REG stores

the priority of the currently served interrupt. Should an interrupt with a higher priority be generated, the processing of the first interrupt is stopped and its priority is stored in the register PRIORITY STACK, and the new interrupt of the higher priority is processed and its priority is stored in the register CURR IRQ PRIORITY REG.

[0008] Once the processing of the interrupt is completed, the previously suspended interrupt is processed provided its priority has remained the highest of the priorities of all pending interrupts. When the processing of any interrupt is completed, its priority is canceled from the stack PRIORITY STACK by a command STACK PUSH/POP of the state machine IRQ SM.

[0009] An important parameter of interrupt controllers is the latency time of interrupts, that is, the time that elapses from the instant of reception of the interrupt in the register INT PENDING REG and the instant in which it is processed. It is always desirable that this time lag be as short as possible. Moreover, an interrupt must be processed within a certain maximum time (dead line) from the instant in which it is loaded in the pending interrupt register, otherwise the application managed by the running program may not function properly.

[00010] To prevent an interrupt from being processed after a pre-established dead line, the priority registers in known controllers are re-programmed at pre-established intervals. The duration of these intervals vary as a function of the register increasing the priority level of interrupts as a function of their latency in the pending interrupt register. In this way, interrupts with a longer latency are given a higher

priority than the interrupts that have just been received, and are eventually processed before their latency reaches the dead line.

[00011] In contrast, the priorities of interrupts stored in the stack PRIORITY STACK are not incremented for preventing an interrupt previously suspended in favor of an incoming interrupt having a higher priority from suspending the processing of the incoming interrupt, and so on. Unfortunately, this technique of re-programming is not very convenient because the recurrent task of re-programming the priority registers burdens the microprocessor, thus slowing execution of the program.

Summary of the Invention

[00012] An object of the present invention is to provide a method for generating interrupt commands in a microprocessor system that overcomes the drawbacks of the known techniques based upon re-programming the priority registers at intervals.

[00013] According to the present invention, it is possible to increase automatically the priority values of the pending interrupts without burdening the microprocessor with such a task. This result is obtained by using counters dedicated to store the effectively used priority values for determining which interrupt is to be processed first. When an interrupt is received, a relative counter containing the respective priority value is incremented at pre-established time intervals by an increment signal.

[00014] This and other objects, advantages and features in accordance with the present invention are provided by a method being implemented in an interrupt

priority control circuit for a microprocessor system. The interrupt priority control circuit includes priority registers for storing priority values associated with respective possible interrupts, one or more pending interrupt registers, a priority logic circuit for generating an interrupt request signal and an internal signal representing the relative priority, and a circuit for processing the interrupt signal and the internal signal that eventually sends to a system processor an interrupt command and an interrupt vector.

[00015] The interrupt control circuit does not require intervention of the system processor to increment the priority values of pending interrupts as a function of their latency because it comprises a plurality of counters coupled to the priority registers that are initialized with the priority values of all types of interrupts to be served. Each counter receives an increment signal of its content and the priority logic circuitry reads from the updated counters the priority value associated to each interrupt. Preferably, the counters are incremented by respective signals because the priorities of each interrupt may vary differently depending on the type of interrupt.

Brief Description of the Drawings

[00016] The different aspects and advantages of the present invention will become more evident through a detailed description of the invention and by referring to the attached drawings, wherein:

[00017] Figure 1 depicts a commonly known interrupt controller according to the prior art;

[00018] Figure 2 shows the general architecture of a system using an interrupt controller according to the

present invention; and

[00019] Figure 3 shows a detailed architecture of an interrupt controller according to the present invention.

Detailed Description of the Preferred Embodiments

[00020] A basic architecture of a system using an interrupt controller INTERRUPT CONTROLLER in accordance with the present invention is depicted in Figure 2. Differently from the prior art controller of Figure 1, the INTERRUPT CONTROLLER block is input also with the signals PRIORITY TRIGGERS that are used to increment the priority of pending interrupts. This is done to prevent the interrupts from remaining unprocessed for a time exceeding a certain pre-established maximum dead line time.

[00021] A preferred embodiment of the interrupt controller of the present invention is depicted in Figure 3. Basically, it differs from the known controller of Figure 1 because of the presence of a plurality of priority counters PRIORITY COUNTERS connected in cascade to the PRIORITY REGISTERS and from which the circuit IRQ MASK AND PRIORITY LOGIC reads the priority values for identifying the highest priority.

[00022] When an interrupt is loaded in the register INT PENDIG REG, the corresponding counter previously loaded with the value stored in the corresponding PRIORITY REGISTER is enabled to be periodically incremented by the respective increment signal of the signals fed to the counters block PRIORITY TRIGGERS. The circuit IRQ MASK AND PRIORITY LOGIC reads from the periodically incremented counters the priority values, identifies which of the pending interrupts has the

highest priority, generates an interrupt request IRQ REQ that is sent to the state machine IRQ SM and an internal signal HIGHEST PRIORITY INT representing the priority of the interrupt INTn that must be served.

[00023] At this time, the counter that stores the priority value relative to the saved interrupt INTn is re-initialized with the original priority value. The count is stopped and the interrupt INTn is canceled from the pending interrupt register.

[00024] The priority values of all the pending interrupts stored in the counters PRIORITY COUNTERS are incremented at pre-established time intervals that may be different from counter to counter by respective increment signals PRIORITY TRIGGERS. In this way the priority of a pending interrupt is automatically incremented via hardware. By so doing, interrupts are eventually served before their latency reaches the dead line time. The controller advantageously frees the microprocessor from the task of re-writing at pre-established intervals the content of the priority registers.

[00025] To manage nested interrupts, the controller may be provided with the registers CURR IRQ PRIORITY REG and PRIORITY STACK, as in the controller of Figure 1. For example, the increment signals fed to the counters may be derived from the clock signal of the controller. They may be generated by frequency dividers or they may be derived from an external clock signal coming from the operating system, or they may be generated by events external or internal to the microprocessor. For example, they may be the result of a comparison or a signal applied to a pin of the device.

[00026] Preferably, each counter is incremented by a respective increment signal to allow the priorities to be differently incremented depending on the type of interrupt to which each of them refers. Preferably, the increment signal of a counter has a period chosen as a function of the allowed maximum latency so that the relative interrupt may reliably reach the top priority value before the time limit elapses. To insure this, the period of the increment signal will be made shorter than the ratio between the allowed maximum latency and the difference between the expected maximum and minimum priority values.